MEDIAN RED ALGORITHM FOR CONGESTION CONTROL

Gonzalo R. Arce, Kenneth E. Barner, and Liangping Ma

Department of Electrical and Computer Engineering
University of Delaware, Newark, DE 19716, USA

ABSTRACT

This paper focuses on the queue size estimation problem in *Random Early Detection* (RED) gateways. Queue size estimation plays a critical role in gateways' packet dropping/marking decision. Conventional RED gateways use exponentially weighted moving averages (EWMA) to estimate the queue size. These IIR filters require very small weights in order to avoid nonlinear instabilities and accommodate transient congestion. Small weights, however, lead to gateways' failure to closely track rapid queue size depletion and thus causes link under utilization. We use *adaptive weighted median* filters for queue size estimation and study the corresponding queue dynamics. Simulation results show that the proposed algorithm provides better stability in queue dynamics, greater network power, less global synchronization, and a fairer treatment to bursty traffic than the RED algorithm.

1. INTRODUCTION

The congestion control mechanisms of Transmission Control Protocol (TCP) are not adequate for the congestion control of the Internet. To complement this, active queue management has been proposed [1] [2]. Among the several approaches for active queue management proposed to date, Floyd and Jacobson's *Random Early Detection* (RED) has been the most widely implemented. The RED algorithm consists of two main tasks: (a) estimation of the average queue size at the gateway, and (b) packet drop decision. The basic idea is to sense impending congestion before it occurs and try to provide feedback to the senders by either marking or dropping their packets, even if space is still available at the queue.

The average queue size estimate in RED is updated at the time of packet arrival k according to:

$$\hat{q}_k = (1 - w) \cdot \hat{q}_{k-1} + w \cdot q_k \qquad (1)$$

where $\hat{q}_k$ is the estimate of the average queue size at time k and q_k is the instantaneous queue size. The queue average estimate in (1) is a first-order IIR filter. As discussed in [1], when a packet arrives into an empty queue, the estimate in (1) is modified taking into account how much time has elapsed since the queue went empty.

The recursive estimate in (1) is not uniformly effective. To prevent nonlinear stabilities in the queue dynamics [3] and to accommodate bursty traffic [1], the parameter w must be set to a very low value[1]. But small-valued w results in severe low-pass queue estimates, which can not track rapid queue size depletion. This leads to unnecessary packet drops/marks and hence link under utilization [4].

To improve the queue size estimation of conventional RED, we present a new queue size estimation algorithm, the Median RED algorithm [4], which is based on weighted median(WM) filtering [5]. This algorithm is effective in tracking rapid drops in queue size and is equally effective in sensing impending congestion trends. In addition, the resulting average queue dynamics do not exhibit instabilities. As a result, the Median RED algorithm provides better stability in the queue dynamics and more efficient network resources allocation than the conventional RED algorithm.

The remainder of this paper is organized as follows. Section 2 reviews the RED algorithm and introduces the Median RED algorithm. Section 3 compares Median RED and RED through analysis and simulations, and draws conclusions.

2. TCP-RED MODEL FOR MULTIPLE FLOWS

A model of RED congestion control at a gateway is depicted in Fig. 1. The system considers M TCP flows sharing a common link with capacity C packets/sec and one way propagation delay D. Throughout this paper, all TCP connections are assumed to be NewReno [6] connections where forward TCP traffic is a one way flow between source S_j and the receiver nodes R_j. The reverse traffic consists of only acknowledgment(ACK) packets. The delay from S_j to the RED Gateway is d_j.

This work was supported in part by the National Science Foundation ITR-ANI Grant 0312851.

[1]Typical values are in the range (0.005, 0.02).

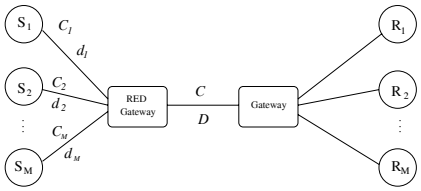

Fig. 1. M TCP flows sharing a bottleneck link.

2.1. Conventional RED Algorithm

Conventional RED gateways utilize the linear filter in (1) for average queue size estimation. The average queue size estimate is then compared to two thresholds, a minimum threshold $q_{\min}$ and a maximum threshold $q_{\max}$ and packets are dropped at a probability given by [1]

$$p(\hat{q}) = \begin{cases} 0 & \text{if } \hat{q}_k < q_{\min}, \\ 1 & \text{if } \hat{q}_k > q_{\max}, \\ \frac{\hat{q}_k - q_{\min}}{q_{\max} - q_{\min}} p_{\max} & \text{otherwise.} \end{cases} \quad (2)$$

By dropping packets, the router triggers a feedback system where TCP traffic sources adjust their transmission rates.

The shortcoming of the linear queue estimator is that if it is configured to be able to smooth out transient increases in queue size then it can not track sudden queue depletion. This is because a linear filter that smoothes out a sudden increase in a signal will also smooth out a sudden decrease, as is shown in Fig. 2. More detailed discussion is in [4].

2.2. Median RED Algorithm

Given the queue size observations $q_k, q_{k-1}, \cdots, q_{k-L}$, the *adaptive Weighted Median* estimate of the average queue size $\hat{q}_k$ is defined as

$$\hat{q}_k = \text{MEDIAN}[q_{k-L+1}, \ldots, q_{k-1}, w(r_k) \diamond q_k] \quad (3)$$

where $\diamond$ is the replication operator defined as $w(r_k) \diamond q_k = \overbrace{q_k, q_k, \cdots, q_k}^{w(r_k) \text{ times}}$, the weight $w(r_k) = K+1-r_k$, and r_k is the rank of q_k among the samples in the observation window. The constant K is typically set to L.

The adaptive WM filter in (3) emphasizes the current observation q_k when q_k is small in the observation window, and de-emphasizes q_k otherwise. Using the adaptive WM queue estimate, the Median RED thus quickly tracks queue depletions while being able to smooth out bursty queue increases, as is shown in Fig. 2. More detailed discussion on the properties of the adaptive WM filter is in [4].

The computational complexity of the adaptive WM filter allows implementations of large window size since, if properly designed, the complexity consists of $O(\log_2 L)$ comparisons, at most L addition and threshold operations [4] [5].

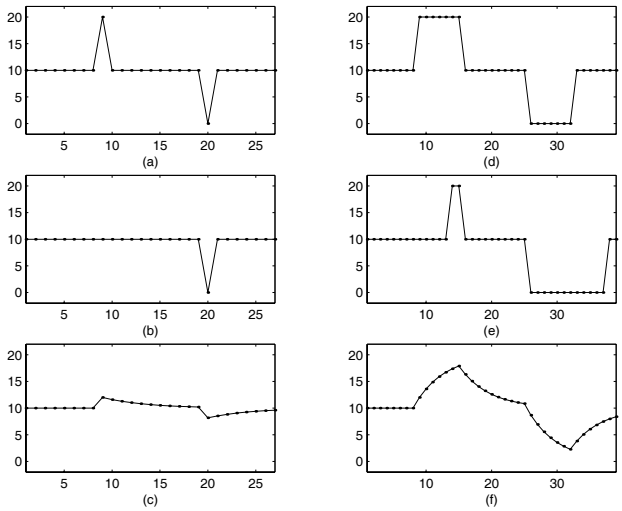

Fig. 2. Responses of Median RED and conventional RED on impulses: (a) the input positive pulse and negative pulse with width 1, the responses of (b) Median RED with window size 11 and of (c) conventional RED with $w = 0.2$. (d)(e)(f) correspond to pulses with width 7.

3. COMPARING MEDIAN RED AND RED

In this section we compare the Median RED algorithm with the conventional RED through analysis and simulations.

3.1. Analytic Results

We study the queue dynamics of Median RED for the network in Fig. 1 with the conventional RED algorithm replaced by the proposed Median RED algorithm. Under the assumptions [3] that
1) the link (with capacity C) between the two gateways is the only bottleneck in the network, and that
2) all M TCP flows are long-lived and have the same round-trip time and maximum window size,
we can derive the following queue dynamics [4]

$$\begin{aligned} \hat{q}_{k+1} &= \text{MEDIAN}(q_{k-L+2}, \ldots, q_k, q_{k+1} \diamond w_{k+1}) \\ &= \text{MEDIAN}(\frac{h}{\sqrt{\hat{q}_{k-L+1} - q_{\min}}}, \ldots, \frac{h}{\sqrt{\hat{q}_{k-1} - q_{\min}}}, \\ &\quad w_{k+1} \diamond \frac{h}{\sqrt{\hat{q}_k - q_{\min}}}) - CR_0, \end{aligned} \quad (4)$$

where $w_{k+1} = L + 1 - r_{k+1}$, R_0 is the fixed part of the round-trip time, λ depends on TCP implementations and it is equal to $\sqrt{3/4}$ for TCP Reno, and h is given by

$$h = M\lambda\sqrt{\frac{q_{\max} - q_{\min}}{p_{\max}}}. \quad (5)$$

If the steady-state average queue estimate $\lim_{k\to\infty} \hat{q}_{k+1}$ exists, the limit, denoted by $\hat{q}_\infty$, is determined by

$$\hat{q}_\infty = \frac{h}{\sqrt{\hat{q}_\infty - q_{\min}}} - CR_0, \tag{6}$$

which has a unique solution

$$\hat{q}_\infty = \frac{g(h,d)}{6} + \frac{2d^2}{3g(h,d)} + \frac{1}{3}(q_{\min} - 2CR_0), \tag{7}$$

where

$$g(h,d) = (108h^2 + 8d^3 + 12h(81h^2 + 12d^3)^{\frac{1}{2}})^{\frac{1}{3}}, \tag{8}$$

and $d = CR_0 + q_{\min}$. It is important to note that the derived steady-state average Median RED queue estimate is the same as that of the conventional RED algorithm [3].

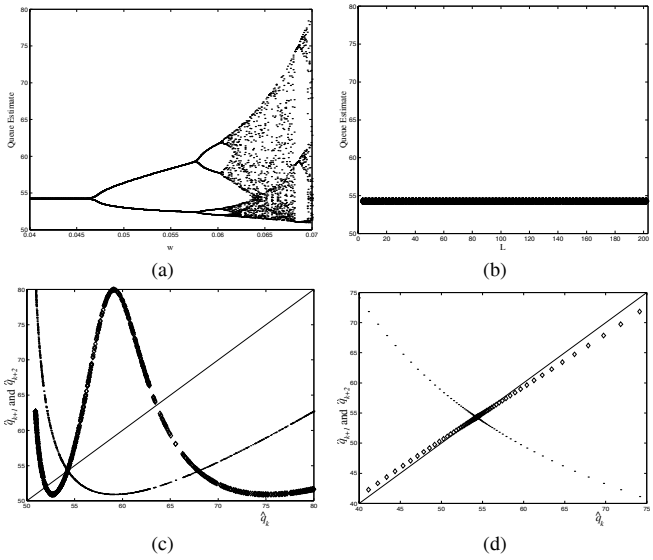

Fig. 3. The steady-state average queue estimates (a) for RED with different weights and (b) for Median RED with different window sizes. The first-return maps (dots) and the second return maps (diamonds) for (c) RED ($w = 0.07$) and for (d) Median RED ($L = 200$). The solid lines are at 45^o.

The steady-state queue estimates can be formally investigated through the use of first–return and second–return maps[2]. The first-return and second-return maps for the RED and Median RED are plotted in Fig. 3 (c) and (d), respectively. Note that the intersections of the 45^o line and the maps give the fixed points [7], or steady state queue estimates. For the RED algorithm, the absolute values of the slopes at the fixed points are all greater than 1, indicating that these fixed points are unstable. In the Median RED case, there is only one fixed point and the absolute value of the slope at this point is less than 1, implying a stable fixed point [7]. This explains the difference in stability between the RED and Median RED algorithms.

[2]For a sequence: $x_1, x_2, \ldots, x_n, \ldots$, its first-return map consists of the following points (x_n, x_{n+1}) for $n = 1, 2, \ldots$ on a plane. And its second-return map is composed of the points (x_n, x_{n+2}) for $n = 1, 2, \ldots$ [7].

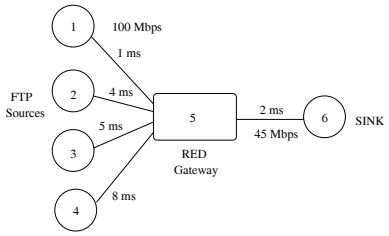

Fig. 4. The simulation network for network power.

The network power is given by [4]

$$P = \frac{\lambda}{(R_0 + \frac{\hat{q}_\infty}{C})^2 \sqrt{\frac{p_{\max}(\hat{q}_\infty - q_{\min})}{q_{\max} - q_{\min}}}}, \tag{9}$$

which suggests that the most effective network resource allocation is attained when $\hat{q}_\infty$ is close to $q_{\min}$. Guidelines for Median RED parameter selection are provided in [4].

3.2. Simulation Results

We now compare Median RED and conventional RED through ns [8] simulations on TCP NewReno [6]. The metrics used for the comparison are network power, global synchronization, and fairness.

We first consider the network power. The simulation network in Fig. 4 has four FTP sources, labeled as nodes 1 through 4, that send data to node 6. The FTP sources start at random in the time interval [0.0, 1.0] seconds and all data packets are 1000 bytes. The maximum window sizes for the FTP sources are 33, 66, 78, and 112 packets. The parameters for the RED gateway are $p_{\max} = \frac{1}{50}$, $w_q = 0.002$, $q_{\max} = 3q_{\min}$ where $q_{\min} = 5, 10, 15$, and 20, and the buffer size of the gateway is $B = 100$ packets. The Median RED parameters are set to correspond with the RED parameters [4]. The adaptive WM filter window sizes are set to $L = 135, 187, 227$ and 257 for $q_{\min}$ of 5, 10, 15 and 20 respectively. All other Median RED parameters are set equal to those for RED. The results for different $q_{\min}$ values are shown in Fig. 5(a), where each data point is the average of 5 random simulations. The figure shows that Median RED achieves network power that is comparable to RED at low throughput and is higher than RED at high throughput.

We next consider global synchronization utilizing the global synchronization index [4], which measures the degree of global synchronization and increases as global synchronization becomes more severe. Figure 5(b) plots the in-

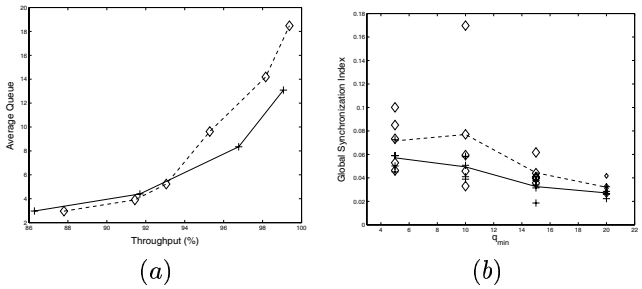

Fig. 5. (a) The average queue size changes with the aggregate throughput for Median RED (solid line with pluses) and RED (dashed line with diamonds); (b) global synchronization indexes for Median RED (pluses) and RED (diamonds) for different values of $q_{\min}$, where the solid lines connect the averages for Median RED, and the dashed line connects the averages for RED.

dexes for different values of $q_{\min}$ for the network in Fig. 4. It is clear that Median RED's average global synchronization index is consistently smaller than that of the RED, indicating less severe global synchronization. This is because Median RED can closely track sudden decreases in queue size, which prevents link under utilization and hence global synchronization [4].

We finally consider fairness to bursty traffic. The simulation network in Fig. 6 has five FTP sources, nodes 1 through 5, which send data to node 7. The maximum window size for FTP sources one through four is 12 packets. The bursty traffic is generated by node 5, which has a much larger RTT than the other sources as well as a smaller maximum window size of 8. Other network parameters are set as $p_{\max} = 0.02$ and $q_{\max} = 2q_{\min}$ with $q_{\min}$ investigated over the range 4 to 14. The gateway buffer size is set to $4q_{\min}$. The throughput of node 5 as a percentage of the aggregate throughput is plotted in Fig. 7. The plots clearly show that Median RED results in a higher and more stable share of bandwidth for bursty traffic. The improved performance of Median RED can again be related to its accurate tracking of depleting queues. If the actual queue is rapidly depleting, RED will not accurately track this trend and if the next round of packets are from the bursty source, random drops will likely occur heavily penalizing the bursty source. In contrast, Median RED accurately tracks such trends allowing the bursty traffic to pass.

3.3. Summary

The analysis and simulations show that the proposed Median RED algorithm yields better performance than the conventional RED algorithm by providing more stability in queue dynamics, more network power, less global synchronization, and more fairness to bursty traffic.

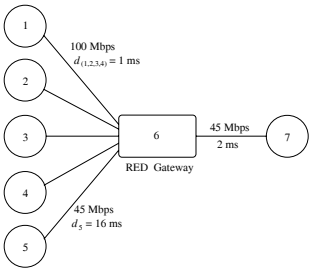

Fig. 6. Simulation network for bursty traffic.

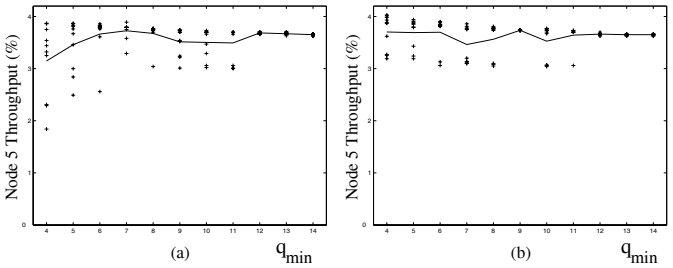

Fig. 7. (a) The throughput of node 5 as a percentage of the total throughput for RED and for (b) Median RED.

4. REFERENCES

[1] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, 1993.

[2] W. Feng, D. D. Kandlur, D. Saha, and K. G. Shin, "A self-configuring RED gateway," in *Proceedings of INFOCOM 99*, San Francisco, CA, March 1999, vol. 3, pp. 1320–1328.

[3] P. Ranjan, E. Abed, and R. La, "Nonlinear instabilities in TCP-RED," in *Proc. of IEEE INFOCOM 2002*, New York City, NY, June 2002.

[4] G. R. Arce, K. E. Barner, and L. Ma, "Red gateway congestion control using median queue size estimates," *IEEE Transactions on Signal Processing*, vol. 51, no. 8, pp. 2149–2164, August 2003.

[5] G. R. Arce, "A general weighted median filter structure admitting negative weights," *IEEE Trans. Signal Processing*, vol. 46, no. 12, pp. 3195–3205, Dec. 1998.

[6] S. Floyd, "RFC 2582: The NewReno modification to TCP's fast recovery algorithm," Apr 1999.

[7] P. Bergé, Y. Pomeau, and C. Vidal, *Order within Chaos*, Hermann, Paris, France, 1984.

[8] http://www.isi.edu/nsnam/ns.